Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Отчёт по лабораторной работе № 3**

**По курсу «Разработка программного обеспечения систем управления»**

**«Декомпозиция программы»**

Выполнил студент группы А-02-20

Аношкин Д.В.

Проверил

Мохов А.С.

Козлюк Д.А.

Василькова П.Д.

Москва 2021

**Цель работы:**

* Уметь структурировать программу при помощи функций.
* Уметь писать модульные тесты.

**Порядок выполнения:**

**Часть 1. Декомпозиция программы функциями**

* Требуется вынести часть кода с вводом чисел в массив в отдельную функцию, которую назовет input\_numbers.

vector<double>

input\_numbers(size\_t count) {

vector<double> result(count);

**for** (size\_t i = 0; i < count; i++) {

cin >> result[i];

}

**return** result;

}

* **С учетом того, что numbers в дальнейшем не изменяется (тип данных можно объявить как const) ввод данных в массив будет происходить так:**

const vector<double> numbers = input\_numbers(count);

В данном случае выгоднее использовать ключевое слово auto, так как оно сокращает часть написанного кода, в итоге ввод данных в массив будет выглядеть так:

const **auto** numbers = input\_numbers(number\_count);

**Объявим функцию find\_minmax()для поиска минимального и максимального элемента массива.**

void find\_minmax(const vector<double>& numbers, double&min, double& max)

{

min = numbers[0];

max = numbers[0];

for (double x : numbers)

{

if (x < min)

min = x;

if (x > max)

max = x;

}

**Создадим функцию, производящую расчет чисел в столбцах гистограммы, назовем ее make\_hixtogram()**

vector<double> make\_histogram(const vector<double>& numbers, double min, double max, size\_t bin\_count)

{

for (double number : numbers)

{

size\_t bin = (size\_t)((number - min) / (max - min) \* bin\_count);

if (bin == bin\_count)

{

bin--;

}

bins[bin]++;

}

}

**Также создадим функцию, выводящую гистограмму на экран, пусть она называется show\_histogram\_text()**

void show\_histogram\_text(vector<double> bins)

{

const size\_t SCREEN\_WIDTH = 80;

const size\_t MAX\_ASTERISK = SCREEN\_WIDTH - 4 - 1;

size\_t max\_count = 0;

for (size\_t count : bins) {

if (count > max\_count) {

max\_count = count;

}

}

const bool scaling\_needed = max\_count > MAX\_ASTERISK;

for (size\_t bin : bins) {

if (bin < 100) {

cout << ' ';

}

if (bin < 10) {

cout << ' ';

}

cout << bin << "|";

size\_t height = bin;

if (scaling\_needed) {

const double scaling\_factor = (double)MAX\_ASTERISK / max\_count;

height = (size\_t)(bin \* scaling\_factor);

}

for (size\_t i = 0; i < height; i++) {

cout << '\*';

}

cout << '\n';

}

}

**Часть 2. Вывод гистограммы как изображения в формате SVG**

**Отработаем цикл модификации и проверки программы на примере пустого изображения, заменим функцию show\_hostogram\_text() на функцию show\_histogram\_svg()**

void svg\_begin(double width, double height)

{

cout << "<?xml version='1.0' encoding='UTF-8'?>\n";

cout << "<svg ";

cout << "width='" << width << "' ";

cout << "height='" << height << "' ";

cout << "viewBox='0 0 " << width << " " << height << "' ";

cout << "xmlns='http://www.w3.org/2000/svg'>\n";

}

void svg\_end()

{

cout << "</svg>\n";

}

void show\_histogram\_svg(const vector<size\_t>& bins)

{

svg\_begin(400, 300);

svg\_end();

}

**Для вывода подписей к столбцам напишем функцию вывода текста в SVG**

void svg\_text(double left, double baseline, string text)

{

cout << "<text x='" << left << "' y='" << baseline << "'>" << text << "</text>";

}

**Для проверки выведем высоту первого столбца гистограммы:**

void show\_histogram\_svg(const vector<size\_t>& bins)

{

svg\_begin(400, 300);

svg\_text(20, 20, to\_string(bins[0]));

svg\_end();

}

**Напишем функцию, выводящую прямоугольник SVG:**

void svg\_rect(double left, double baseline, size\_t width, double height)

{

cout << "<rect x ='" << left << "' y ='" << baseline << "' width ='" << width << "' height ='" << height << "' />\n";

}

**Заменим "магические" числа на константные переменные с понятным названием в функции show\_histogram\_svg()**

void show\_histogram\_svg(const vector<size\_t>& bins)

{

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);

svg\_text(TEXT\_LEFT, TEXT\_BASELINE, to\_string(bins[0]));

svg\_rect(TEXT\_WIDTH, 0, bins[0] \* BLOCK\_WIDTH, BIN\_HEIGHT);

svg\_end();

}

**Реализуем вывод гистограммы при помощи SVG:**

void show\_histogram\_svg(const vector<size\_t>& bins)

{

const auto IMAGE\_WIDTH = 400;

const auto IMAGE\_HEIGHT = 300;

const auto TEXT\_LEFT = 20;

const auto TEXT\_BASELINE = 20;

const auto TEXT\_WIDTH = 50;

const auto BIN\_HEIGHT = 30;

const auto BLOCK\_WIDTH = 10;

svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);

double top = 0;

for (size\_t bin : bins) {

const double bin\_width = BLOCK\_WIDTH \* bin;

svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));

svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT);

top += BIN\_HEIGHT;

}

svg\_end();

}

**Добавим значения параметров по умолчанию функции svg\_rect().**

void svg\_rect(double left, double baseline, size\_t width, double height, string stroke = "black", string fill = "black")

**Реализуем для программы масштабирование, максимальная ширина каждой корзины 350. Сделаем коммит, отправим его на GitHub.**



**Часть 3. Модульное тестирование**

**Перенесем функцию find\_minmax() в histogram.h, а также добавим подключение библиотеки векторов и стандартного пространства имен.**

#pragma once

#include <vector>

using namespace std;

**Необходимо подключить файл histogram.h (в самом начале файла main.cpp)**

#include "histogram.h"

**Добавим к проекту файл реализации .cpp. Также подключим там histogram.h**

**Перенесем в histogram.cpp определение функции find\_minmax()**

void find\_minmax(const vector<double>& numbers, double& min, double& max)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers) {

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

**Создадим новый проект типа lab03-test.**

**Добавим к проекту histogram.cpp и Header.h, а также файл релизации test.cpp.**

Файл test.cpp представляет собой полноценную тестирующую программу, в которой подключен тестируемый модуль, библиотека с assert()

#include "histogram.h"

#include <cassert>

void

test\_positive() {

double min = 0;

double max = 0;

find\_minmax({ 1, 2, 3 }, min, max);

assert(min == 1);

assert(max == 3);

}

int

main() {

test\_positive();

}

**Убедимся, что программа завершается без ошибок, то есть тест проходит.**

**Добавим новые тесты на разные случаи:**

* массив из трех отрицательных чисел;
* массив из трех одинаковых чисел;
* массив из одного числа.

void test\_negative()

{

double min = 0;

double max = 0;

find\_minmax({ -1, -2, -3 }, min, max);

assert(min == -3);

assert(max == -1);

}

void test\_constant() {

double min = 0;

double max = 0;

find\_minmax({ 99,99,99 }, max, min);

assert(min == 99);

assert(max == 99);

}

void one\_number()

{

double min = 0;

double max = 0;

find\_minmax({ 1 }, min, max);

assert(min == 1);

assert(max == 1);

}

**Убедимся, что все тесты проходят.**

**Попробуем написать тест для пустого массива, он не пройдет.**

void test\_empty()

{

double min = 0;

double max = 0;

find\_minmax({}, min, max);

assert(min == 0);

assert(max == 0);

}

**Для того, чтобы этого избежать добавим проверку на наличие элементов в массиве в функции find\_minmax()**

void find\_minmax(const vector<double>& numbers, double& min, double& max)

{

if (numbers.size() != 0)

{

min = numbers[0];

max = numbers[0];

for (double number : numbers) {

if (number < min) {

min = number;

}

if (number > max) {

max = number;

}

}

}

}

**Пересоберем программу и убедимся в том, что она не выдает никаких ошибок.**

**Сделаем коммит со всеми измененными файлами и отправим его на GitHub.**

**Индивидуальное задание.**

Вариант 2

Задавать автоматически яркость заливки каждого столбца гистограммы в градациях серого в зависимости от высоты столбца. Чем больше столбец, тем темнее заливка.

* **Добавим функцию,которая задает яркость заливки столбца в зависимости от его высоты.**
* string brightness (size\_t bin,size\_t max\_count)
* {
* int pr;
* string s="#";
* if (max\_count == 0) return "Error";
* else {
* pr=10 - (bin \* 9) / max\_count;
* for(int i=0;i<3;i++)
* {
* if (pr > 9) pr = 9;
* s += to\_string(pr);
* }
* return s;
* }}
* **Подключим ее в функцию show\_histogram\_svg()**
* void show\_histogram\_svg(const vector<size\_t>& bins)
* {
* const auto IMAGE\_WIDTH = 400;
* const auto IMAGE\_HEIGHT = 300;
* const auto TEXT\_LEFT = 20;
* const auto TEXT\_BASELINE = 20;
* const auto TEXT\_WIDTH = 50;
* const auto BIN\_HEIGHT = 30;
* const auto BLOCK\_WIDTH = 10;
* const size\_t MAX\_ASTERISK = IMAGE\_WIDTH - TEXT\_WIDTH;
* size\_t max\_count = 0;
* for (size\_t count : bins) {
* if (count > max\_count) {
* max\_count = count;
* }
* }
* const bool scaling\_needed = (max\_count \* BLOCK\_WIDTH) > MAX\_ASTERISK;
* double scaling\_factor;
* if (scaling\_needed)
* {
* scaling\_factor = static\_cast<double>(MAX\_ASTERISK) / (max\_count \* BLOCK\_WIDTH);
* }
* else {
* scaling\_factor = 1;
* }
* svg\_begin(IMAGE\_WIDTH, IMAGE\_HEIGHT);
* double top = 50;
* for (size\_t bin : bins) {
* const double bin\_width = BLOCK\_WIDTH \* bin\*scaling\_factor;
* svg\_text(TEXT\_LEFT, top + TEXT\_BASELINE, to\_string(bin));
* string fill;
* fill=brightness(bin,max\_count);
* svg\_rect(TEXT\_WIDTH, top, bin\_width, BIN\_HEIGHT,"black", fill);
* top += BIN\_HEIGHT;
* }
* svg\_end();
* }

**Проверим работоспособность программы, убедимся в отсутстви ошибок. Сделаем коммит, отправим изменения на GitHub.**